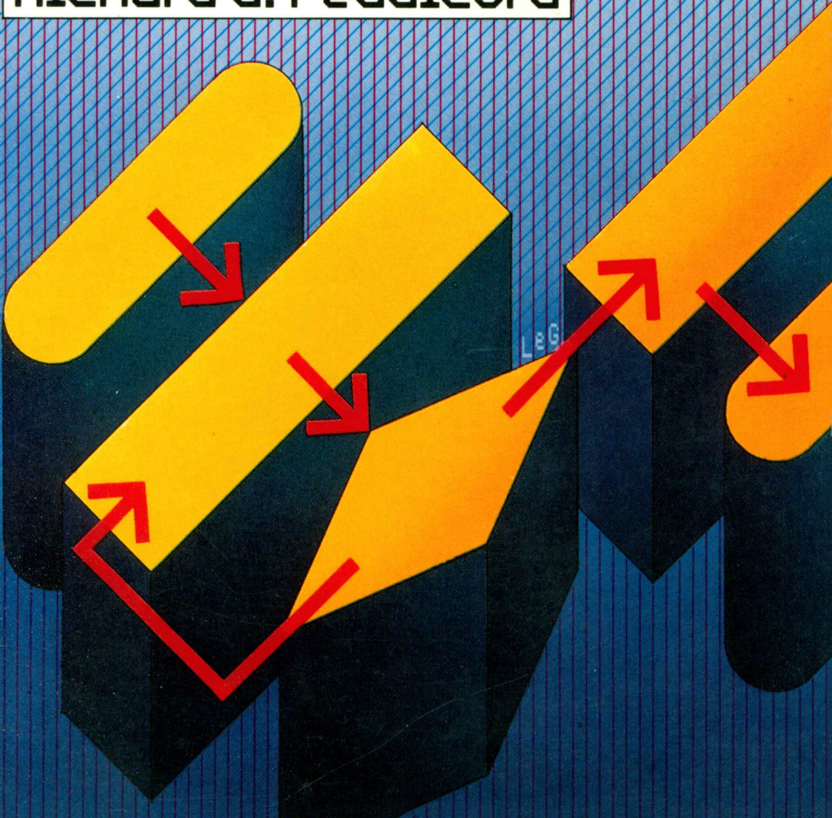
PLAZA & JANES
P & J
EDITORES

Plaza & Janés · Informática

BASIC

Iniciación al lenguaje de programación más difundido

Richard G. Peddicord



PLAZA & JANES

P & J

EDITORES

Plaza & Janés · Informática

Richard G. Peddicord

BASIC

Iniciación al lenguaje de programación más difundido

PLAZA & JANES EDITORES, S. A.

Título original: Understanding Basic
Traductor: Domingo Santos
Portada de «Design team München», cedida por Wilhelm Goldman Verlag, München
Primera edición: Noviembre, 1985

Virgen de Guadalupe, 21-33
Esplugues de Llobregat (Barcelona)

Printed in Spain
Impreso en España
ISBN: 84-01-4010-1
Depósito Legal: B. 37.293 - 1985
Impreso por Grafos, S. A.
Paseo de Carlos I, 157
(Barcelona)

INTRODUCCION

Está usted a punto de introducirse en el lenguaje de programación de ordenadores llamado BASIC. El BASIC es un lenguaje exactamente igual que el inglés, pero es mucho más sencillo. Para aprender a utilizar el BASIC necesitará aprender unas cuantas palabras. Cada palabra será una palabra inglesa, pero su significado en BASIC es único. Necesitará saber usted exactamente cuándo y cómo utilizar cada palabra, de modo que pueda facilitar instrucciones correctas a un ordenador.

Darle instrucciones a un ordenador se le llama *programar*. Un programa le dice exactamente al ordenador lo que tiene que hacer. Cada tarea del ordenador, no importa lo sencilla que sea, debe hallarse indicada en un programa, o la tarea no será realizada. Es más, el ordenador seguirá las instrucciones del programa con toda precisión, incluidos los errores. Como resultante de ello, una correcta programación requiere una atención meticulosa a los detalles.

¿Qué es un *ordenador*? Es una máquina diseñada para ayudarle a resolver problemas relativos al tratamiento y procesado de la información. Puede utilizar usted los ordenadores para que realicen la mayor parte del trabajo rutinario de la oficina, para que lleven la contabilidad de su negocio y sus asuntos financieros, para realizar cálculos complicados, para mantenerse informado de los hechos importantes, e incluso para

entretenerse y desafiar a sus amigos con una serie de juegos.

Lo más interesante de los ordenadores es que son muy rápidos. Una vez se hallan correctamente programados para producir los resultados que se desean, funcionarán con seguridad y sin quejas.

Para comprender de qué elementos está formado un ordenador –o, más correctamente, un sistema informático–, piense en un equipo estéreo. Los componentes habituales de un sistema hi-fi estéreo son: 1), un tocadiscos o una platina; 2), un receptor o amplificador; y 3), un juego de altavoces. Alguien seriamente interesado en la hi-fi puede que tenga también: 4), una unidad de grabación en cinta magnetofónica; 5), una grabadora a cassettes; y 6), unos auriculares.

Un sistema informático típico se compone de una manera similar. Como mínimo, ha de tener: 1), una unidad de entrada de datos; 2), una unidad central de proceso; y 3), una unidad de visualización de datos. La entrada de datos puede efectuarse a través de una *lectora de fichas* y la visualización a través de una impresora. En la mayor parte de los sistemas informáticos, y especialmente en los *microordenadores*, datos e instrucciones pueden entrarse y los resultados pueden visualizarse a través de un solo componente, llamado un *terminal de pantalla*. Un terminal de pantalla posee un teclado como el de una máquina de escribir para entrar los datos e instrucciones, y una pantalla parecida a la de una televisión para visualizar éstos y los resultados.

La parte principal o "cerebro" del ordenador es la *unidad central de proceso* (*Central Processing Unit*: CPU). La CPU efectúa todo el procesado de los datos, realiza los cálculos, toma las decisiones según las instrucciones del programa, y activa la entrada de datos y las unidades de visualización.

Pero, como en los sistemas estéreo, a un sistema informático pueden añadírsele más cosas. Los sistemas informáticos, especialmente los grandes, pueden poseer memorias *periféricas* tales como: 1), unidades

de cinta magnética; 2), unidades de cinta a cassette o cartucho, *hard-disks* o discos duros, discos flexibles (*diskettes*), u otras unidades auxiliares de almacenamiento; y la CPU puede poseer una *interface* (es decir, un elemento de interconexión) con: 3), tableros gráficos, *plotters* de dibujo automático, terminales con *joysticks* (es decir palancas de mando para moverse libremente por la pantalla) y/o lápices ópticos, o con prácticamente cualquier tipo de dispositivo de laboratorio, industrial o de oficina.

En este Manual, va a ser llevado usted a un breve "tour dirigido" a una parte de la ciencia de los ordenadores y del proceso de datos... comprendiendo finalmente cómo pueden ser programados en BASIC los ordenadores.

ACERCA DE ESTE MANUAL

Este Manual es una introducción al lenguaje BASIC para aquellos que no se han visto enfrentados antes al BASIC. Su material está diseñado para ser utilizado junto con un terminal, a fin de que pueda usted ejecutar de una manera práctica los ejemplos antes de seguir adelante.

El material cubierto le permitirá a usted escribir programas de gestión de empresas, de tamaño y complejidad medios, cubriendo *series* de una y dos dimensiones (es decir, vectores y matrices).

La tabla que sigue a continuación indica el subconjunto de elementos del BASIC que es estudiado.

INSTRUCCIONES BASICAS

DATA	LET
DIM	NEXT
END	PRINT, MAT PRINT
FOR	PRINT USING
GOTO	READ, MAT READ
IF ... THEN	REM
INPUT	STOP
IMAGE	

COMANDOS DEL SISTEMA

BYE	RUN
LIST	SAVE/UNSAVE
NEW	
OLD	

EMPEZANDO

El lenguaje de programación BASIC es uno de los más fáciles de aprender, y con mucho el más popular entre los ordenadores personales. Muchas empresas están cambiando al BASIC para poder utilizar programas que se venden comercialmente, y cada día se están efectuando nuevas mejoras en el lenguaje BASIC.

Necesitará usted tener acceso durante unas cuantas horas a un terminal que emplee el BASIC, y disponer de alguien que le enseñe cómo conectarse, o hacer "LOG ON". Escuelas primarias, escuelas secundarias, universidades, academias de informática, empresas, y muchos particulares poseen ordenadores que trabajan con BASIC, de modo que le resultará fácil tener acceso al BASIC sin tener que comprar ningún equipo.

La mayor parte de los microordenadores y modernos sistemas de gran tamaño utilizan *terminales* para *entrar datos* y *programar*. Un terminal consiste en un *teclado* y una *superficie visora* (ya sea una *impresora hard-copy* o una *pantalla CRT**). Los terminales con impresora incorporada son útiles cuando desea usted estudiar con posterioridad sus operaciones con la máquina desde que se conectó al sistema. El papel es caro, sin embargo.

* El CRT (*Cathode Ray Tube*, Tubo de Rayos Catódicos) es un tipo de visualización similar a la pantalla de los aparatos normales de televisión.

La mayoría de los terminales son sin embargo CRT, o terminales de pantalla. La mayor parte de estos terminales almacenan varias páginas de operaciones en su memoria. Si desea ver usted una transacción anterior, pulsa la tecla de *Scrolling** hasta que el dato deseado aparezca ante su vista.

Hay que hacer notar que no todos los teclados son exactamente iguales, a pesar de que básicamente adaptan dos formatos según la disposición del teclado alfabético. Sus caracteres, y su disposición en el teclado, pueden diferir de terminal a terminal. Sin embargo, en general, todos los terminales poseen como mínimo las siguientes teclas:

BARRA ESPACIADORA
RETORNO DE LINEA
TECLADO ALFABETICO Y NUMERICO
TECLA DE CONTROL (CTRL)
TECLA DE MAYUSCULAS
TECLAS DE PARADA Y ESCAPE
BORRADO/RETROCESO
SIGNOS DE PUNTUACION HABITUALES
(),.<>=

* Dicha tecla sirve para visualizar líneas anteriores o posteriores a las que aparecen en la pantalla.

SUS PRIMEROS PROGRAMAS

Teclee este programa en su terminal. Después de cada línea, pulse la tecla de retorno (return) y aguarde a que aparezca el carácter "*" antes de empezar a teclear una nueva línea. Si comete usted un error de escritura, pulse la tecla return y empiece de nuevo (con el mismo número de línea).

> Esos son números de línea. Vea pág. 17 para una explicación de su uso.

```
❶  * 5   REM PROGRAMA FUNCIONA
❷  *10   PRINT "FUNCIONA"
❸  *20   END
```

❶ **Instrucción REM (*Remark*: observación).** Le dice al ordenador que ignore esta línea. Es utilizada para comentarios que usted desee hacer para su propio uso y para el de otras personas que lean su programa.

❷ **Instrucción PRINT (Imprimir).** PRINT hará que el mensaje o dato que sigue sea impreso o aparezca en la pantalla. Cualquier texto/mensaje entre comillas será impreso exactamente igual a como se ha entrado.

❸ Esto le dice dónde termina su programa.

Teclee RUN y observe la pantalla. Obtendrá lo siguiente:

OUTPUT

```
*RUN
 FUNCIONA
*
```

La máquina estará aguardando a que escriba usted un carácter.

Practique el uso de su terminal entrando y procesando el siguiente programa:

```
* 5   REM PROGRAMA POEMA
*10   PRINT "LAS ROSAS SON ROJAS"
*20   PRINT "LAS VIOLETAS SON AZULES"
*30   PRINT "SI NOSOTROS PODEMOS"
*40   PRINT "APRENDER BASIC"
*50   PRINT "ENTONCES USTED TAMBIEN"
*60   END
```

Teclee RUN, y observe la pantalla. Obtendrá este resultado:

```
*RUN
 LAS ROSAS SON ROJAS
 LAS VIOLETAS SON AZULES
 SI NOSOTROS PODEMOS
 APRENDER BASIC
 ENTONCES USTED TAMBIEN
*
```

LOS NUMEROS DE LAS LINEAS

En BASIC, tiene que proporcionar usted un número de línea delante de cada instrucción. Este número de línea obligatorio tiene varias finalidades:

1. Puede referirse usted a cada instrucción por su número. Esto permite correcciones o borrados fáciles, y le da la posibilidad de dirigirse en cualquier momento a cualquier punto de su programa.
2. Si desea añadir usted una instrucción entre otras dos instrucciones ya entradas, todo lo que necesita hacer es darle a la nueva instrucción un número de línea entre los números de línea de las otras dos instrucciones. Antes de que el programa sea ejecutado o listado, todas las instrucciones serán puestas en orden creciente.

INPUT

❶
```
*10  X=2
*20  PRINT X,Y
*15  Y=3
```

❶ **Instrucción de asignación.** Asigna el valor de la derecha a la variable de la izquierda. En este caso, tras la asignación, X contendrá el número 2. Este ejemplo puede ser escrito también como

```
*10 LET X=2
```

Si tras teclear una línea determinada (supongamos la línea 20 del ejemplo anterior) descubre usted que ha cometido un error, puede corregirlo en la línea siguiente insertando un número intermedio entre el de las dos líneas que le correspondan, (en ese caso puede ser el 15). Cuando pulse RUN, las instrucciones se ordenarán en número creciente antes de ejecutarse el programa:

OUTPUT

```
*RUN
 2                3
*
```

Si quiere efectuar usted una comprobación, la instrucción LIST hará que el ordenador le muestre lo que lleva escrito hasta el momento. En la pantalla, las instrucciones aparecerán por orden:

INPUT

```
*LIST
```

OUTPUT

```
10   X=2
15   Y=3
20   PRINT X,Y
*
```

SALVANDO SUS PROGRAMAS

Un vez ha conseguido usted un programa operativo, puede que desee usted conservarlo. Para almacenar un programa en el sistema, piense en un nombre único *que usted pueda recordar* (utilice sus iniciales en algún lugar de él). Tras el carácter de preparado (el asterisco, "*"), teclee SAVE y pulse return. El sistema le preguntará por el nombre bajo el cual desea almacenar usted su programa. Teclee el nombre (cópielo en el margen de este libro), pulse *return*, y *aguarde pacientemente* a que el sistema indique que está preparado de nuevo, o aparezca algún mensaje de error.

Para comprobar que su SAVE ha funcionado realmente, teclee NEW, seguido por return. El comando NEW, en este caso, borrará lo que tenga en aquel momento en su área de trabajo. Luego teclee LIST, seguido por return. No obtendrá ningún listado, solamente el preparado, puesto que ahora está trabajando usted con un nuevo espacio de trabajo en el que no tiene aún ningún programa.

Para reclamar lo que ha salvado, teclee OLD, seguido por el nombre del programa que utilizó usted al salvarlo (no olvide pulsar return tras cada orden). Luego teclee LIST. Su programa aparecerá, si usted lo salvó bien.

Si ya no necesita más un programa almacenado, teclee UNSAVE seguido por el nombre del archivo. Desaparecerá de la memoria de la máquina. Cuando uti-

lice usted el BASIC en tiempo compartido con otros usuarios, utilice el UNSAVE para eliminar sus programas siempre que no los necesite ya más. (Compruebe siempre el manual de su ordenador. Las palabras utilizadas por los comandos del sistema pueden variar; por ejemplo, UNSAVE puede ser SCRATCH.)

LAB 1. ARITMETICA SIMPLE

Compruebe que su sistema funciona correctamente efectuando un poco de aritmética. Empiece lentamente, con tareas como:

INPUT

```
 5  REM SUMA
10  PRINT 2 + 7
RUN
```

OUTPUT

```
9
```

INPUT

```
 5  REM RESTA
10  PRINT 2.7 - 8.342
RUN
```

OUTPUT

```
-5.642
```

Símbolos aritméticos

(No todas las versiones del BASIC utilizan esos símbolos)

Potenciación	** ó ↑	Suma	+
Multiplicación	*	Resta	—
División	/		

INPUT

```
 5  REM SUMA Y DIVIDE
10  PRINT(3.6+4,2)/(-8.762)
RUN
```

Prioridad.
El ordenador efectuará normalmente primero la potenciación, luego la multiplicación y la división (avanzando de izquierda a derecha), y luego la suma y la resta (avanzando de izquierda a derecha). Lo que se halle dentro de los paréntesis, sin embargo, será ejecutado siempre primero.

OUTPUT

```
-.8902
```

INPUT

```
 5  REM UTILIZACION DE VARIABLES
10  X=3.5
20  Y=-18.3009
30  PRINT (X-2+Y**2)/(X-Y)**2
RUN
```

OUTPUT

```
.7078
```

Practique salvando y llamando sus programas.

COMAS, PUNTOS Y COMAS Y TABULADOR

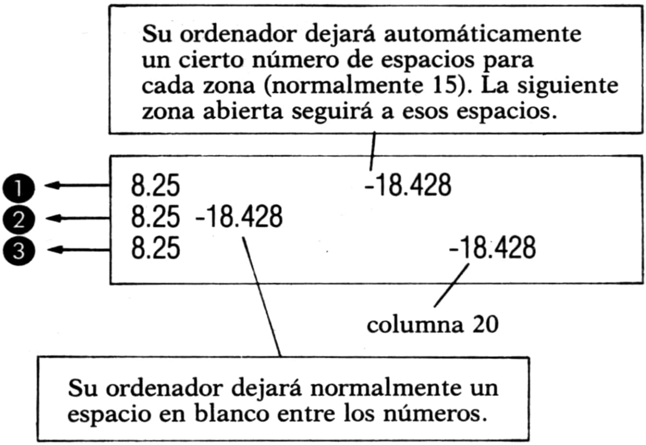
Los siguientes signos de puntuación son utilizados entre campos impresos para indicar espaciado entre esos campos. Los principales signos de control son:

, (la coma)	significa...	tabulación a la siguiente zona abierta (generalmente hay cinco zonas en una línea).
; (el punto y coma)	significa...	juntar.
TAB(N)	significa...	saltar a la columna N.

Si alguno de estos signos aparece al final de una línea impresa, hace que la siguiente instrucción PRINT empiece en la misma línea. Esto puede resultar útil en organizar su output de modo que tenga un buen aspecto. Observe un ejemplo en la página siguiente.

Experimente con , ; y TAB.

```
80  X=8.25
90  Y=-18.428
100  PRINT X,Y                produce  ——→ ❶
100  PRINT X;Y                produce  ——→ ❷
100  PRINT X,TAB(20),Y        produce  ——→ ❸
```



VARIABLES ALFANUMERICAS Y VARIABLES NUMERICAS

Una *variable numérica* es un nombre variable que almacena un número. En la mayoría de las versiones del BASIC, un nombre variable numérico consiste en una letra, o en una letra seguida por un dígito.

Nombres de variables numéricas válidos:	Nombres de variables numéricas no válidos:
A, X, D7, H4	A&, $B, 95, TOOLONG

Este nombre puede ser válido en algunos BASIC. Compruebe para ver si es válido en el suyo.

Para asignar un valor a un número variable, escriba

```
10   X = 100
```

Una *variable de cadena* o variable alfanumérica es un nombre variable que almacena una hilera de caracteres (como por ejemplo un nombre). En la mayor par-

te de las versiones del BASIC, un nombre variable alfanumérico consiste en una sola letra seguida por un signo de dólar.

Nombres de variables alfanuméricas válidos:	Nombres de variables alfanuméricas no válidos:
A$, W$, G$	$A, $$, X, 9X, JAJA

Este nombre es válido para una variable numérica, pero no válido para una variable alfanumérica.

Para asignar un valor a una variable alfanumérica, escriba

```
15    A$ = "SMITH"
```

El ordenador almacenará todo lo que esté entre las comillas.

INSTRUCCIONES INPUT

La instrucción INPUT obtiene datos de su terminal durante la realización de su programa. Por ejemplo:

INPUT

```
10  PRINT "ENTRA UN NUMERO"
20  INPUT X
30  PRINT "GRACIAS"
40  END
```

hace que aparezca un signo de interrogación.

OUTPUT

```
*RUN
ENTRA UN NUMERO
?
243
GRACIAS
*
```

En este punto usted entra 243.

Adelante, inténtelo. Observe que la máquina está esperando a que le teclee usted algo. Tecléele un número y pulse return. Vea lo que ocurre si no teclea usted un número correcto.

Cuando le teclea usted un número correcto y pulsa return, la máquina sitúa el número que usted le ha tecleado en la localización llamada X, y luego se traslada a la siguiente instrucción de su programa. En este caso, no hay mucho que hacer. El carácter de preparado aparecerá pronto.

LAB 2. INTERES COMPUESTO

Podemos ilustrar la utilización de un input e instrucciones de asignación utilizando un problema de interés compuesto. En este problema, utilizamos un input para la cantidad que deseamos invertir, el capital, *c*, en pesetas; luego otro para el tipo anual de interés, *i*, en forma decimal; luego otro para el número de años que deseamos esperar, *n*. Utilizamos la fórmula

$$s = c(1 + i)^n$$

para calcular el nuevo capital, *s*, tras *n* años de inversión. El listado que sigue muestra el programa y el resultado de su ejecución.

(Advertencia: aunque en la actualidad un número cada vez mayor de los ordenadores que se comercializan en España vienen ya preparados para el alfabeto y los signos de puntuación castellanos, puede que su ordenador posea aún un teclado inglés. Si es así, observará que los signos "¿" y "¡", así como la letra "Ñ", no figuran enél. En ese caso, simplemente prescinda de estos signos (que verá que aparecen solamente en instrucciones REM), y sustituya la Ñ por otra letra.

También es probable (vea pág. 23) que su ordenador no le acepte los dos signos normalmente utilizados para la exponenciación. Algunos ordenadores admiten solamente la ↑ . Compruébelo. En ese caso, le bastará sustituir en sus programas el signo de exponenciación ** por la ↑ .

Y ahora pruebe el programa:

INPUT

```
100  REM LAB 2. INTERES COMPUESTO
110  PRINT "¿CUANTO QUIERE INVERTIR?"
120  INPUT C
130  PRINT "¿CUAL ES EL TIPO ANUAL DE INTERES?"
140  INPUT I
150  PRINT "¿CUANTOS AÑOS QUIERE ESPERAR?"
160  INPUT N
170  S=C*(1+I)**N
180  PRINT "TRAS";N;"AÑOS TENDRA";S;"PESETAS"
230  END
```

OUTPUT

```
*
RUN
¿CUANTO QUIERE INVERTIR?
?
10000
¿CUAL ES EL TIPO ANUAL DE INTERES?
?
.17
¿CUANTOS AÑOS QUIERE ESPERAR?
?
8
TRAS 8 AÑOS TENDRA 35114.5 PESETAS
```

LA INSTRUCCION IF...THEN

En el ejemplo que hallarán en la página siguiente, si el número entrado es el 999, entonces el programa se dirigirá a la línea 100, imprimirá «VALE», y se detendrá. Si no, simplemente irá a la siguiente línea del programa, en este caso la línea 40, que inicia de nuevo el ciclo.

Hay una forma más general de la instrucción IF que se halla disponible en los microordenadores más nuevos. Le permite a usted especificar una lista de cosas que hacer en caso de que se presente una cierta condición, en vez de solamente una orden de ir a un número de instrucción. Puesto que la instrucción general IF no siempre se halla disponible, no la utilizaremos en este Manual.

Recuerde esta regla acerca del IF: si la condición es cierta, el programa salta a la instruccón cuyo número aparece después de THEN; si la condición es falsa, el programa prosigue con la siguiente instrucción. Una instrucción IF puede comprobar otras condiciones además de la igualdad. Los símbolos <> emparejados significan *no igual a*, <= significan *menor que o igual a*, y >= significan *mayor que o igual a*.

Veamos el ejemplo:

Instrucciones GO TO. Simplemente indican al ordenador que salte a una línea determinada.

INPUT

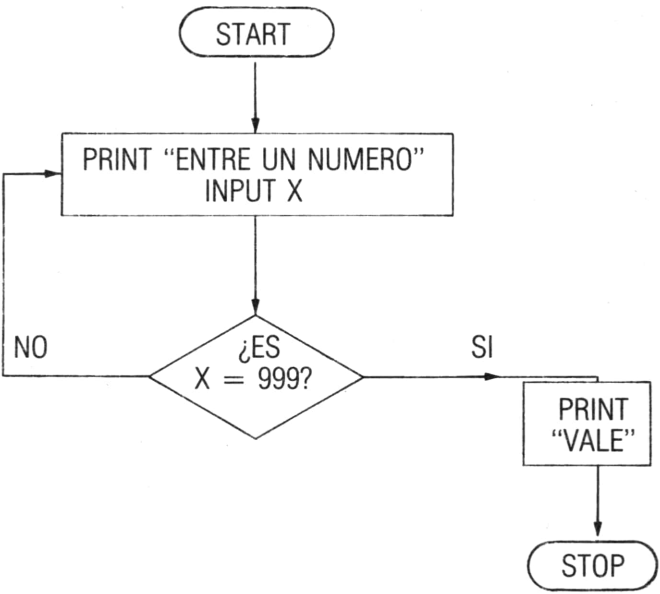
```
  5 REM ****EJEMPLO 4****
 10 PRINT "ENTRE UN NUMERO"
 20 INPUT X
 30 IF X = 999 THEN 100
 40 GOTO 10
100 PRINT "VALE"
120 END
```

OUTPUT

```
*RUN
ENTRE UN NUMERO
?
14
ENTRE UN NUMERO
999
VALE
```

LA INSTRUCCION IF...THEN
(continuación)

Esto es un **organigrama**. Nos muestra la lógica de un programa.



Podemos utilizar la instrucción IF para añadir flexibilidad a nuestro programa del interés compuesto.

INPUT

```
100  REM LAB 2. INTERES COMPUESTO
110  PRINT "¿CUANTO QUIERE INVERTIR?"
120  INPUT C
130  PRINT "¿CUAL ES EL TIPO ANUAL DE INTERES?"
140  INPUT I
150  PRINT "¿CUANTOS AÑOS QUIERE ESPERAR?"
160  INPUT N
170  S=C*(1+I)**N
180  PRINT "DESPUES DE";N;"AÑOS TENDRA";S;"PESETAS"
190  PRINT "DESEA CONTINUAR (S/N)"
200  INPUT A$
210  IF A$="S" THEN 110
220  PRINT "FIN DEL TRABAJO"
230  END
```

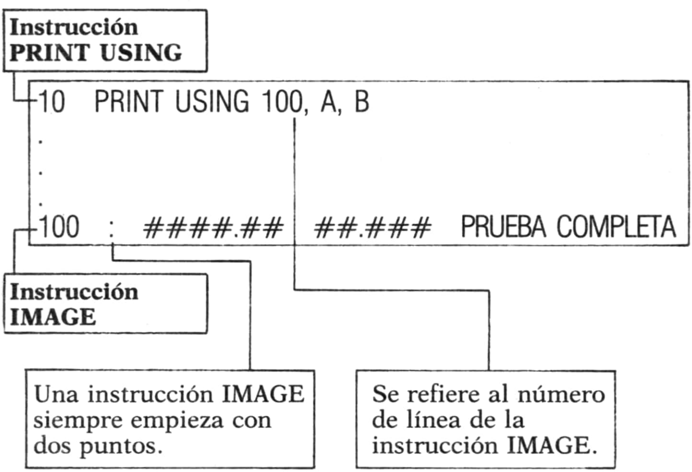
OUTPUT

```
*
RUN
¿CUANTO QUIERE INVERTIR?
?
10000
¿CUAL ES EL TIPO ANUAL DE INTERES?
?
.17
¿CUANTOS AÑOS QUIERE ESPERAR?
?
8
DESPUES DE 8 AÑOS TENDRA 35114.5 PESETAS
```

```
DESEA CONTINUAR (S/N)
?
N
FIN DEL TRABAJO
```

PRINT USING

Hay otra forma de controlar su espaciado entre informaciones impresas además de las comas y los puntos y comas. En la instrucción PRINT USING, nos referimos a una instrucción IMAGE separada que nos muestra cómo imprimir las variables listadas en la instrucción de impresión. Por ejemplo, el par de instrucciones:



dará lugar a la siguiente línea impresa (suponiendo que A = 23.9637 y B = 8.5527):

```
23.96      -8.552      PRUEBA COMPLETA
```

En general, el compilador BASIC reconoce una hilera de signos "#" –que en algunos ordenadores se sustituye por el signo de la libra esterlina, por lo que en lenguaje informático se lo conoce también como "signo de libra"–, o una hilera de signos "#" con una coma decimal intercalada*, como un grupo de posiciones de *caracteres* donde imprimir una sola variable, ya sea una variable numérica o una variable alfanumérica.

Si no especifica usted sitio suficiente en su hilera de signos "#", y empieza a imprimir un campo que es más largo que el espacio disponible, los caracteres extra de la derecha se perderán. Si intenta imprimir usted un número que es demasiado grande para el espacio disponible, de todos modos, normalmente se imprimirán asteriscos.

Otros caracteres en una instrucción IMAGE, aparte el signo "#", son impresos tal como están escritos, y el primer carácter de la línea impresa empieza inmediatamente después de los dos puntos. En una instrucción IMAGE, los espacios en blanco no son ignorados.

*. No olvidemos que en inglés, y en consecuencia en muchos ordenadores, la coma decimal se representa por un punto.

EJERCICIO

Escriba una instrucción PRINT USING que produzca el siguiente output:

```
col. 1                    col. 20
|                         |
8.25                      -18.428
```

RESPUESTA

```
 80   X = 8.25
 90   Y = -18.428
100   PRINT USING 200,X,Y
200   :##.##                 ###.###
```

LOS BUCLES FOR...NEXT

A menudo es necesario ejecutar repetidamente un bloque de *instrucciones*. Para ello se utilizan un par de instrucciones especiales. La primera, una instrucción FOR, actúa al principio del bloque que ha de ser repetido. La segunda, una instrucción NEXT, actúa al final del bloque.

La instrucción FOR tiene la forma general:

Dice cuantas veces tiene que ser repetido el bloque de instrucciones.

FOR [índice del bucle] = [valor de inicio] TO [valor de final] STEP [incremento del valor]

una variable

Si el incremento es 1, puede eliminarse.

La instrucción NEXT tiene la forma general:

NEXT [índice del bloque]

Esto le dice al ordenador que añada el incremento al índice y vuelva a la instrucción FOR

Algunos ejemplos:

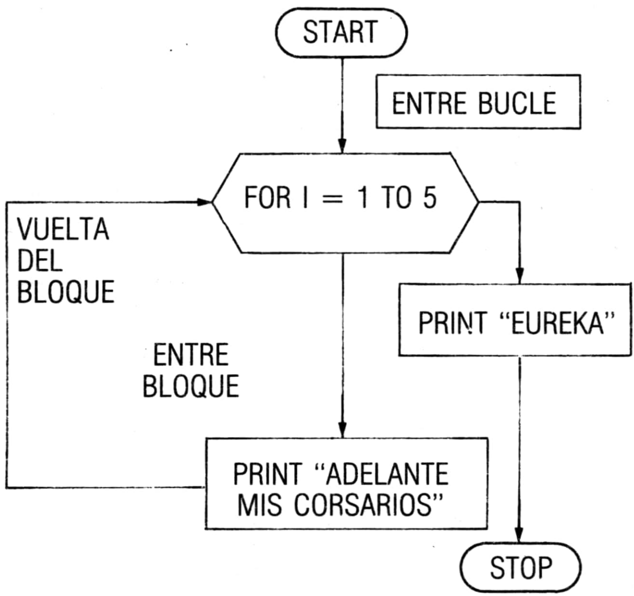
```
FOR I = 1 to 10 STEP 2
.
.
.
NEXT I
```

Esto será repetido 5 veces.

```
FOR J = 2 to 11
.
.
.
NEXT J
```

Esto será repetido 10 veces.

Pruébelo. Ejecute este programa:



```
10    REM BUCLES FOR NEXT
20    FOR I = 1 TO 5
30    PRINT "ADELANTE MIS CORSARIOS"
40    NEXT I
50    PRINT "        ¡EUREKA!"
60    END
```

Tiene que conseguir el siguiente output:

```
ADELANTE MIS CORSARIOS
ADELANTE MIS CORSARIOS
ADELANTE MIS CORSARIOS
ADELANTE MIS CORSARIOS
ADELANTE MIS CORSARIOS
        ¡EUREKA!
```

LEYENDO DATOS

Si tiene usted un conjunto fijo de datos y desea procesar su programa con ellos, el BASIC le permite agrupar los datos en una o más instrucciones DATA, como en:

```
10  REM     EJEMPLO DATOS
20  DATA 4,6,2
30  DATA -6,8
```

El sistema recoge todas sus instrucciones DATA separadas en una instrucción más larga, empezando con la instrucción DATA con el número de línea más pequeño. Cada vez que es ejecutada una instrucción READ en su programa, la siguiente entrada disponible en las instrucciones DATA es utilizada para proporcionar los datos para la instrucción READ. Así, si añadimos al programa de arriba las instrucciones:

```
40  READ A,B
50  READ C,D,E
```

se producirían los siguientes pasos. En la línea 40 el valor 4 irá a A, el valor 6 a B. En la línea 50, el 2 es leído en C, -6 pasa a D, y 8 pasa a E.

Si no hay suficientes datos para suministrar a todas las instrucciones READ, se imprimirá un mensaje

al respecto. Para ver lo que es este mensaje, añada estas líneas y procese el programa.

```
60    PRINT A,B,C,D,E
70    READ F
```

Esto producirá un mensaje de error como: OUT OF DATA AT LINE 70.

¿Y si desea usted leer los datos alfanuméricos? Entonces necesitará utilizar variables alfanuméricas en la instrucción READ para almacenar los datos, y la cadena de caracteres tiene que situarse normalmente entre comillas en la instrucción DATA. Por ejemplo,

```
130    DATA 1,134.89,"HILTON HAWAII"
.
.
.
270    READ T,A,V$
```

cadena de datos

variable alfanumérica

LAB 3. CUENTA DE GASTOS

Considere el siguiente problema hipotético. Tiene usted 10 instrucciones DATA, cada una de las cuales contiene el tipo T, la cantidad C, y el concepto M$ referente a un solo gasto de un breve viaje de negocios a Hawaii. Escriba un programa para leer las 10 instrucciones DATA y procesarlas de acuerdo con las siguientes pautas:

1. Calcule un rembolso R de la siguiente manera:
   Si T=1, entonces R=100% de la cantidad C.
   Si T=2, entonces R=80% de la cantidad C.
   Si T=3, entonces R=20% de la cantidad C.
   Si T=cualquier otra cantidad, entonces escriba "número tipo equivocado" y el número, y pase a leer la siguiente instrucción de datos.

2. Imprima el tipo T, el concepto M$, la cantidad C, y el rembolso R, tal como ha sido calculado en 1.

3. Cuando hayan sido procesadas las 10 instrucciones, imprima "todas las instrucciones procesadas" y pare.

Intente utilizar la instrucción PRINT USING para el espaciado de su output, y utilice un bucle FOR... NEXT para controlar la lectura de las instrucciones DATA. Intente obtener el siguiente output:

OUTPUT

```
 *
 RUN
 REEMBOLSO DE CUENTA DE GASTOS
TIPO  MOTIVO                        CANT.   REEMB.
  1   HILTON HAWAII                134.89   134.89
  3   JOE'S CASINO                  44.32     8.86
  2   CHEN'S QUICK FOOD              4.89     3.91
  1   BETTY'S BREAKFAST NOOK         5.88     5.88
  1   TAXI                           3.66     3.66
  3   DIAMOND HEAD TOURS             1.45     0.29
  1   HILTON HAWAII                 46.89    46.89
  2   BUD'S BOOKS                   33.77    27.02
  2   TAXI                          14.39    11.51
  3   JOE'S CASINO                  57.99    11.60
 TODAS LAS INSTRUCCIONES PROCESADAS
```

El programa sigue a continuación. Observe que dentro de las instrucciones DATA hemos incluido el concepto con dobles comillas. En algunos sistemas esto no es necesario.

```
100  REM   LAB 3. CUENTA DE GASTOS
110  REM   SIGUEN 10 INSTRUCCIONES DE DATOS,
120  REM   DANDO CADA UNA, POR ORDEN,
130  REM   TIPO, CANTIDAD Y MOTIVO
140  DATA 1,134.89,"HILTON HAWAII"
150  DATA 3,44.32,"JOE'S CASINO"
160  DATA 2,4.89,"CHEN'S QUICK FOOD"
170  DATA 1,5.88,"BETTY'S BREAKFAST NOOK"
180  DATA 1,3.66,"TAXI"
190  DATA 3,1.45,"DIAMOND HEAD TOURS"
```

```
200  DATA 1,46.89,"HILTON HAWAII"
210  DATA 2,33.77,"BUD'S BOOKS"
220  DATA 2,14.39,"TAXI"
230  DATA 3,57.99,"JOE'S CASINO"
240  REM       *****FIN DE LOS DATOS*****
250  PRINT "REEMBOLSO CUENTA DE GASTOS"
260  PRINT "TIPO   CONCEPTO          CANT.   REEMB."
270  FOR I = 1 TO 10
280  READ T,A,M$
290  IF T<>1 THEN 310
300  R=A
310  GOTO 380
320  IF T<>2 THEN 340
330  R=.80*A
340  GOTO 380
350  IF T<>3 THEN 370
360  R=.2*A
370  GOTO 380
380  PRINT "NUM. TIPO EQUIVOCADO EN LINEA DATA",T
390  REM   PROGRAMA PROSIGUE AQUI
400  PRINT USING 400,T,M$,A,R
410   #  ##############   ###.##   ###.##
420  NEXT I
430  PRINT "TODAS LAS INSTRUCCIONES PROCESADAS"
440  END
```

CONCEPTO DE SERIES

En el mundo real, la mayor parte de los números no son más que elementos individuales de un grupo de números, por ejemplo, la temperatura cada hora, o la categoría o el número de la seguridad social de los empleados de una compañia. No resulta práctico darle a cada valor separado un nombre distinto, a menos que automaticemos el procesado de esos nombres. Aquí es donde entran las series.

Hay dos tipos de series utilizados en este Manual, las series unidimensionales, y las series bidimensionales (tratadas en la página 61).

Una serie 1D o unidimensional es una hilera de celdillas, conteniendo cada una de ellas un valor. He aquí un ejemplo:

	1	2	3	4	5	6	7	8	9	10
Q	5	3	1	–2	8	4	–9	0	0	7

En el ejemplo mostrado aquí, el nombre de la serie es Q, y hay diez celdillas en la serie. Cada celdilla tiene un número.

Si deseamos referirnos al número de la celdilla 7, por ejemplo, escribiremos Q(7), como en

```
50   X = Q(7)
```

La instrucción de asignación de arriba toma el valor actual almacenado en la séptima celdilla de Q y lo deposita en X.

Si deseamos situar un cierto número, digamos 3.14, en la celdilla 2 de Q, diremos

```
60   Q(2) = 3.14
```

En casi todas las versiones del BASIC hay una celdilla extra, la celdilla 0, que está disponible. La olvidaremos. Si utiliza usted una serie con menos de 10 celdillas no tiene que darle dimensión. Si, en cambio, necesita más de 10 celdillas, entonces, antes de empezar a utilizar la serie, debe colocar una instrucción de dimensión diciendo su nombre y lo grande que es.

numérico = nombre de la serie

```
100   DIM A(15), X(15), M2(300)
```

número de elementos en la serie

Las series pueden tener los mismos nombres que las variables numéricas.

El siguiente programa construye una serie conteniendo los 20 primeros números primos, y luego los imprime.

En este ejemplo, en la línea 30 el ordenador asigna primero el valor 1 a K. En la línea 40 asigna el valor 2 x 1 - 1 a N(1) (no olvide que multiplica antes de restar). En la línea 50, K es incrementado a 2 y el bucle prosigue.

N(1)	N(2)	N(3)	....	N(20)
1	2	3	......	39

```
10    REM NUMEROS PRIMOS
20    DIM N(20)
30    FOR K = 1 TO 20
40    N(K) = 2 * K - 1
50    NEXT K
60    FOR I = 1 TO 20
70    PRINT N(I)
80    NEXT I
90    END
```

OUTPUT

```
 1
 3
 5
 7
 9
 .
 .
 .
 .
39
```

EJERCICIO

Cambie la instrucción 70 a:

```
70    PRINT N(I),
```

Note la coma al final. ¿Qué es lo que hará? (Vea página 25).

Procese el programa para averiguarlo.

LAB 4. ANALISIS DE VENTAS

Escriba un programa en BASIC para procesar un conjunto de instrucciones DATA. La primera instrucción DATA contiene el número de los artículos que llevan los representantes que deben ser procesados. Cada instrucción siguiente contiene el nombre de un representante y las ventas de cada uno de 5 artículos. Lea cada instrucción DATA y sume las ventas de los 5 artículos para obtener el total de ventas de ese representante. Al mismo tiempo, vaya sumando las ventas de cada artículo para conseguir al final el total de ventas de dicho artículo. Imprima el nombre, las ventas de los 5 artículos, y el total de ventas del representante. Luego lea la siguiente instrucción DATA.

Cuando todas las instrucciones DATA hayan sido procesadas, imprima las ventas totales para cada uno de los 5 artículos; luego imprima "trabajo terminado" y pare.

Sugerencias: utilice 2 series unidimensionales, cada una con 5 elementos, una para los datos y una para los totales. Utilice un bucle FOR...NEXT para controlar el número de grupos de datos leidos, y otro bucle FOR... NEXT para controlar la suma.

Estudie el siguiente programa hasta que comprenda cómo funciona. Luego éntrelo y procéselo. Observe que las instrucciones image 380 y 390 se hallan situadas contiguas, pese a que son utilizadas en instrucciones print que se hallan muy separadas (la 200 y la 370).

Hay una razón natural para ello: resulta más fácil establecer el formato de la cabecera (línea 200) cuando puede usted compararlo con el formato de la línea de datos (impresa en 370).

Su output debe ser algo parecido a esto:

```
 ANALISIS DE VENTAS

              VENTAS POR ARTICULO

REPRESENT.     1   2   3   4   5 TOTAL VENTAS

PEREZ          3   0   5   4  10 22
LOPEZ          1   6  10   2   3 22
PUIG           0  10   3   6   6 25

POR ART        4  16  18  12  19

 TRABAJO TERMINADO
```

CLAVE PARA EL LAB. 4

```
110  REM    ***** LAB 4 ANALISIS DE VENTAS *****
120  DATA 3
130  DATA "LOPEZ",3,0,5,4,10
140  DATA "PEREZ",1,6,10,2,3
150  DATA "PUIG",0,10,3,6,6
160  REM    FIN DE LOS DATOS
170  REM    EMPIEZA PROGRAMA
180  PRINT "ANALISIS DE VENTAS"
190  PRINT "                VENTAS POR ARTICULO"
200  PRINT USING 400, "REPRESENTANTES",
                    1,2,3,4,5,"TOTAL VENTAS"
210  PRINT
220  READ N
230  REM    N ES EL NUMERO DE REPRESENTANTE
240  REM    LA SERIE 1D S CONTIENE LAS VENTAS
                           DE LOS 5 ARTICULOS
250  REM    LA SERIE 1D D CONTIENE LOS TOTALES
                                  POR ARTICULO
```

Esto indica el número de representantes.

```
260  DIM S(5),D(5)
270  FOR I=1 TO N
280  READ N$,D(1),D(2),D(3),D(4),D(5)
290  REM   INICIALIZAR TOTAL VENTAS A CERO
300  T=0
310  FOR J=1 TO 5
320  REM   TOMAR LAS VENTAS DE CADA
330  REM   ARTICULO Y SUMARLAS AL TOTAL
340  REM   DE VENTAS POR REPRESENTANTE Y
350  REM   SUMARLAS TAMBIEN AL TOTAL
360  REM   DE VENTAS POR ARTICULO
370  T=T+D(J)
380  S(J)=S(J)+D(J)
390  NEXT J
400  PRINT USING 390,N$,D(1),D(2),D(3),D(4),D(5),T
410  :####   ##   ##   ##   ##   ##      ##
420  :####    #    #    #    #    #    #####
430  NEXT I
440  PRINT
450  PRINT USING 390,"POR ARTICULO",S(1),S(2),S(3),S(4),S(5)
460  PRINT
470  PRINT "TRABAJO TERMINADO"
480  END
```

Esta instrucción sitúa a cero el valor inicial.

Una buena idea incluir una instrucción de dimensión, pero innecesaria puesto que el número de componentes es inferior a 10.

SERIES BIDIMENSIONALES O MATRICES

En una serie bidimensional (2D), tenemos un campo rectangular de celdillas:

NUM. FILA **NUM. COLUMNA**

	1	2	3	4
1	100	20	3	1
2	20	.02	25	18
3	10	1.0	16	15

F

Este valor será asignado a F(3,2).

Para obtener un valor de una serie 2D empleamos dos números, el de la fila y el de la columna, en este orden. Por ejemplo, para transferir el valor en F(3,2) a una variable llamada X, escribimos:

```
15 X = F(3,2)
```

Efectúe 3 instrucciones DATA para construir la serie de arriba, cada una con una fila de datos (4 números).

```
100    REM 2D EJEMPLO
110    DATA 100,20,3,1
120    DATA 20,.02,25,18
130    DATA 10,1.0,16,15
140    DIM F(3,4)
```

Es siempre necesaria una instrucción de dimensión para las series bidimensionales.

Luego añada el código necesario para leer e imprimir los datos, almacenándolos en el bloque F.

```
150    FOR I = 1 TO 3
160    FOR J = 1 TO 4
170    READ F(I,J)
180    PRINT F(I,J),
190    NEXT J
200    PRINT
210    NEXT I
```

Si su sitema posee MAT READ y MAT PRINT sus programas serán más cortos.

```
150    MAT READ F
160    MAT PRINT F
```

La línea 180 muestra el uso de una coma al final de la línea. Esto hace que los siguientes valores sean impresos en la misma línea, hasta que se produzca una instrucción PRINT que no termine con una coma o un punto y coma. En el ejemplo de arriba, la instrucción 200 hace que la línea quede completa.

Si la tabla en su pantalla concuerda con los datos de la primera tabla de la página 61, está usted preparado para emprender el último trabajo de laboratorio.

LAB 5. ANALISIS DE COSTES DE TRANSPORTE

En este trabajo de laboratorio final, programaremos un procedimiento que seleccione el mejor método de transporte (AVION, TREN o CAMION) para una carga en particular entre dos almacenes. En nuestro modelo, hay 5 almacenes:

1. Seattle, Washington
2. San Diego, California
3. Salt Lake City, Utah
4. Cleveland, Ohio
5. Wheeling, West Virginia

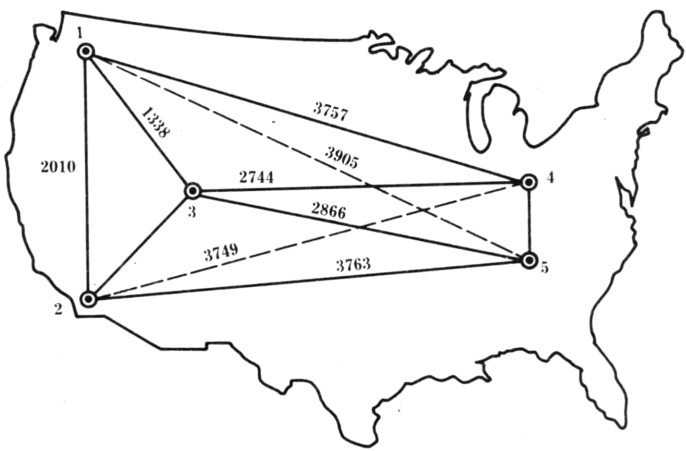
Cada una de las cargas que tiene usted que procesar consiste en una sola instrucción DATA con 4 valores:

NUMERO DE ORDEN	ALMACEN DE ORIGEN
N	O
ALMACEN DE DESTINO	PESO EN TONELADAS
D	P

El conjunto de datos para las pruebas iniciales son los siguientes:

N	O	D	P
2345	2	5	.07
2346	5	3	3.5
2347	1	2	24.2
2348	4	3	0.77
2349	5	4	1.98
2350	3	1	6.97
2351	1	4	0.22

Nuestro procedimiento lee y procesa una carga (instrucción DATA) cada vez. Los cuatro valores son leídos en N,O, D, y P; la distancia entre origen y destino es tomada de una tabla bidimensional, llamada K.



ALMACEN DE DESTINO

ALMACEN DE ORIGEN	1	2	3	4	5
1	0	2010	1338	3757	3905
2	2010	0	1222	3749	3763
3	1338	1222	0	2744	2866
4	3757	3749	2744	0	242
5	3905	3763	2866	242	0

Esta tabla K, o Tabla Kilométrica, está almacenada en una serie bidimensional de dimensión y nombre K(5,5). La distancia entre origen 2 y destino 5 es K(2,5), y contendrá el valor 3763 durante todo nuestro proceso.

Supongamos que K1=K(O,D) sea la distancia entre origen y destino.

Una vez determinada la distancia K1, es computado el coste de envío por cada uno de los tres métodos. Esto requiere otra tabla bidimensional a la que poder acudir, esta vez una Tabla de Fletes F.

	Límite peso cambio de tarifa 1	**Coste por debajo 2**	**Coste por encima 3**
Avión 1	0.25	7.50	5.28
Tren 2	33.00	0.63	0.42
Camión 3	3.0	1.32	0.34
Unidades	**Toneladas**	**Dólares por km/ton**	**Dólares por km/ton**

Para cada método I (I = 1,2,3), compute el coste C(I) como sigue:

$$C(I)= \begin{cases} K1*P*F(I,2) \text{si} P<F(I,1) \\ K1*(F(I,1)*F(I,2)+(P-F(I,1))*F(I,3)) \text{si} P>=F(I,1) \end{cases}$$

Encuentre qué valor de I posee el menor costo C(I). Imprima el número de embarque, origen, destino, distancia (K1), y el método de menor costo, seguido por su costo. Asegúrese de incluir una copia impresa de sus tablas de kilómetros y fletes, a fin de poder comprobar sus datos. Un ejemplo de un output para este trabajo aparece en la página de al lado.

Sugerencias: Utilice MAT READ y MAT PRINT para meter los datos de los kilómetros dentro de la serie K. Haga lo mismo con los datos de fletes. A medida que vaya desarrollando el programa, procéselo siempre que pueda, y sálvelo antes de seguir adelante. Conseguir que el output tenga una apariencia exactamente correcta requiere paciencia.

```
***** LAB 5. ANALISIS DE COSTES DE TRANSPORTE *****

TABLA KLMS.
     0        2010     1338     3757     3905
  2010           0     1222     3749     3763
  1338        1222        0     2744     2866
  3757        3749     2744        0      242
  3905        3763     2866      242        0

TABLA FLETES
   .25         7.5     5.28
    33         .63      .42
     3        1.32      .34

 NUM  ORIG DEST   PESO    DIST  MET      COSTE
 2345   2    5    0.07    3763   2      165.95
 2346   5    3    3.50    2866   2     6319.53
 2347   1    2   24.20    2010   3    22447.68
 2348   4    3    0.77    2744   2     1331.11
 2349   5    4    1.98     242   2      301.87
 2350   3    1    6.97    1338   2     5875.29
 2351   1    4    0.22    3757   2      520.72

FIN DATOS; TRABAJO TERMINADO
```

En las dos páginas siguientes se halla el programa que produjo este output.

```
 10  REM
 20  REM
100  REM     ***** LAB 5. ANALISIS DE COSTES
                           DE TRANSPORTE *****
110  PRINT "***** LAB 5. ANALISIS DE COSTES
                           DE TRANSPORTE *****"
120  REM     SIGUEN DATOS PARA TABLA KILOMETROS
130  DATA 0,2010,1338,3757,3905
140  DATA 2010,0,1222,3749,3763
150  DATA 1338,1222,0,2744,2866
160  DATA 3757,3749,2744,0,242
170  DATA 3905,3763,2866,242,0
180  REM     FIN DATOS TABLA KILOMETROS
190  REM     PONER DATOS EN MATRIZ
200  DIM K(5,5)
210  MAT READ K
220  PRINT "TABLA KILOMETROS"
230  MAT PRINT K
240  REM     SIGUEN DATOS PARA TABLA FLETES
250  DATA .25,7.50,5.28
260  DATA 33.00,0.63,0.42
270  DATA 3.0,1.32,.34
280  REM     FIN DATOS TABLA FLETES
290  REM     PONER DATOS EN MATRIZ
300  DIM F(3,3)
310  MAT READ F
320  PRINT "TABLA FLETES"
330  MAT PRINT F
340  REM     SIGUEN DATOS A PROCESAR
350  DATA 2345,2.5,.07
360  DATA 2346,5,3,3.5
370  DATA 2347,1,2,24.2
380  DATA 2348,4,3,0.77
390  DATA 2349,5,4,1.98
```

```
400   DATA 2350,3,1,6.97                  señala el final
410   DATA 2351,1,4,0.22                  de los datos.
420   DATA 0,0,0,0  ---------------------------'
430   REM    FIN DE LOS DATOS (TIENE NUMERO
                                 ORDEN CERO)
440   PRINT USING 670
450   REM    INICIAR INSTRUCCIONES PROCESO DATOS
460   READ N,O,D,P                    busca el final
470   IF N=0 THEN 700  -------------  de los datos.
480   REM    BUSCA DISTANCIA
490   K1=K(O,D)
500   FOR I=1 TO 3
510   REM    COMPUTAR COSTE PARA METODO I
520   IF P>=F(I,1) THEN 550
530   C(I)=K1*P*F(I,2)
540   GOTO 560
550   C(I)=K1*(F(I,1)*F(I,2)+(P-F(I,1))*F(1,3))
560   NEXT I
570   REM    HALLAR METODO DE MENOR COSTO
580   T=C(1)
590   FOR J=2 TO 3
600   IF C(J)>=T THEN 630
610   T=C(J)
620   M=J
630   NEXT J
640   REM    IMPRIMIR DATOS EMBARQUE Y SOLUCION
650   PRINT USING 660,N,O,D,P,K1,M,C(M)
660   :  ####    #     #     ##.##
                 ####      #   #####.##
670   :  NUM  ORIG  DEST  PESO  DIST  MET  COSTE
680   REM    LEER NUEVO EMBARQUE
690   GOTO 460
700   PRINT "FIN DATOS; TRABAJO TERMINADO"
710   END
```

Hay un ligero error en el programa anterior que puede usted corregir tras procesarlo tal como está. En la línea 620, cualquier valor de J que produzca un costo inferior es almacenado en la variable M. Olvidamos fijar el valor de M a 1 tras haber inicializado la localización temporal T.

Para fijar el programa, inserte 585 M = 1 y procéselo de nuevo. ¿Hay algún cambio en el output?

ADONDE IR DESDE AQUI

Hay varios lugares a los que puede usted desear ir desde aquí. Si desea aprender más acerca del BASIC, encuentre un buen libro de texto que le resulte fácil de comprender, y trabaje una gran cantidad de ejemplos y ejercicios. No hay sustituto para el trabajo con un terminal, y muy pronto descubrirá lo fácil y poderoso de usar que es el BASIC.

Si desea aprender lenguajes similares, el FORTRAN 77 puede ser un buen lenguaje digno de aprender. El BASIC y el FORTRAN se han ido haciendo cada vez más similares con el paso de los años, y los mejores rasgos de uno han sido adoptados por el otro. En la actualidad, sin embargo, el BASIC es con mucho el lenguaje más popular para microordenadores.

Una vez sepa usted un poco de BASIC, podrá computar gráficos. Varios microordenadores personales pueden relizar excelentes gráficos a color, y hay disponible una amplia variedad de software. Vaya a su tienda local de ordenadores y pida una demostración de gráficos.

La música computerizada es otra utilización popular de los microordenadores, y muchos fabricantes de BASIC han añadido a sus versiones un juego de instrucciones que producen sonidos. Hay disponibles programas que memorizan y reproducen canciones.

GLOSARIO

Acceso remoto Equipo periférico conectado al procesador central pero físicamente distante de él.

Acrónimo Una palabra formada por la primera o primeras letras de varias palabras. Muchos términos informáticos son acrónimos.

ALGOL *ALGOrithmic Language*, lenguaje algorítmico. Uno de los primeros y más importantes desarrollos en los lenguajes de programación de alto nivel. Utilizado principalmente para describir algoritmos y como la elección preferida para presentar, impresos, conceptos de programación de ordenadores.

Alfanumérico Cualquier carácter legible para un ser humano y representable en la memoria de un ordenador. (*Ver* Carácter.)

Algoritmo Una secuencia precisa de pasos que definen una computación específica. También es un método general de solución a un problema computable.

Almacenamiento, almacén Memoria.

Análisis de sistemas El estudio de sistemas, específicamente los sistemas informáticos orientados a la gestión de negocios.

APL *A Programming Language,* Un Lenguaje de Programación. Un poderoso lenguaje de programación de alto nivel orientado a las operaciones con series o matrices.

Archivo Una colección de registros.

Array *Ver* Serie.

ASCII *American Standard Code for Information Interchange,* Código Estándar Americano para el Intercambio de Información. Una forma estándar mediante la cual son codificados los caracteres en los ordenadores (principalmente en los no-IBM). (*Ver también* EBCDIC.)

Assembler *Ver* Ensamblador.

Base El número de dígitos utilizados en un sistema numérico. El sistema numérico decimal utiliza la base 10. El sistema binario utiliza la base 2.

Base de datos El juego completo de datos que puede ser utilizado para tomar decisiones y efectuar cálculos y tabulaciones. Su importancia para la organización que la ha compilado es medida por el hecho de cuántas de las operaciones de dicha organización dependen de un frecuente acceso a una base de datos constantemente actualizada.

BASIC El lenguaje de programación a alto nivel más sencillo y fácil de aprender. Es un acrónimo de *Beginner's All-purpose Symbolic Instruction Code,* Código de Instrucción Simbólica de Utilidad General para Principiantes.

Bit *bi*nary digi*t*, dígito binario, un 0 o un 1.

Blank El carácter visible que representa un solo espacio vertical en cualquier cadena.

BNF *Backus Naur Form*. Un "metalenguaje", es decir, un lenguaje o anotación utilizado para definir y describir lenguajes de programación. Llamado así en honor a John Backus (el principal artífice del FORTRAN) y Peter Naur (el responsable de los Comunicados ALGOL-60).

Booleano Un valor que puede ser o *verdadero* o *falso*. Referido también como **lógico**. Llamado así en honor al matemático inglés George Boole. En español se le llama también **boleano**.

Bucle Repetición (de una lista de instrucciones localizadas entre el principio y el final del bucle)

Bug (Textualmente "bicho", "insecto"). Un error que impide que un programa funcione, o lo haga correctamente. Para librar a un programa de tales errores, los programadores realizan un ritual al que denominan "debugging" (literalmente desinsectación) o depuración.

Byte Una secuencia de bits, normalmente 8 bits. Dos bytes forman una "media-palabra" en la memoria del ordenador; y 4 bytes, una palabra (32 bits) completa. El tamaño de una "palabra" en un ordenador, sin embargo, depende de su diseño.

C Un lenguaje algorítmico desarrollado en los Laboratorios Bell.

Campo Un conjunto de espacios, dentro de un registro, en donde cada espacio está dedicado a un propósito específico. Por ejemplo, el registro de un empleado contiene los campos del nombre y del domicilio.

Carácter Un símbolo representable en la memoria de un ordenador. Puede ser alfabético (como una letra), numérico (como un dígito), o alfanumérico especial (como un signo de puntuación o cualquier otro símbolo especial)

Cinta Ya sea cinta de papel o magética: un medio de almacenamiento para datos o instrucciones.

COBOL *COmmon Business-Oriented Language*, Lenguaje Orientado a la Gestión de Negocios. El lenguaje de programación más ampliamente utilizado para aplicaciones comerciales.

Código Una forma específica para representar información y para manipular símbolos.

Codificación Escribir un programa mediante un código, es decir, con un lenguaje específico de programación, y, normalmente, en hojas especiales llamadas "formularios de codificación"

Compilador Un programa especial que traduce el programa completo del usuario (escrito en un lenguaje de programación de alto nivel) al lenguaje máquina. El programa del usuario es referido como "código fuente", que el compilador convierte en "código objeto", hileras de ceros y unos (instrucciones binarias) que pueden ser ejecutadas directamente por la CPU.

Core Memory Memoria central primaria.

CPU *Central Processing Unit*, Unidad Central de Proceso. La parte del ordenador que ejecuta el programa. Llamada también *procesador central* o *unidad central*.

CRT *Cathode Ray Tube*, Tubo de Rayos Catódicos. La pantalla visora (parecida a la pantalla de un televisor) que forma parte de un terminal de ordenador.

Cursor Un subrayado o un cuadrado, a menudo parpadeante, que señala la posición en la cual será introducido el siguiente carácter en una pantalla CRT.

Chip Una pieza rectangular cortada de una placa de silicio. Cada componente LSI es denominado un "chip".

Dato Información codificada.

Debugging Literalmente, *desinsectación*. El proceso de eliminar los errores de programación que impiden que un programa funcione, o le impiden funcionar correctamente. (*Ver* Bug.) Se le llama también depurar.

Dígitos Símbolos utilizados para representar números en un sistema en particular. En el sistema binario, hay dos dígitos, llamados bits: 0 y 1. En el sistema decimal, hay diez dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, y 9.

Disco Un dispositivo de memoria auxiliar (secundaria) parecido a un disco fonográfico.

Diskette Un pequeño y delgado objeto circular revestido magnéticamente, que casi siempre va protegido por un envoltorio o funda cuadrado, y es utilizado como un medio de almacenamiento de bajo costo para sistemas informáticos pequeños, especialmente microordenadores. Es llamado también *floppy disk*, disco blando.

EBCDIC *Extended Binary Coded Decimal Interchange Code*, Código de Intercambio Decimal Codificado en Binario Extendido. Un código utilizado en IBM y otros ordenadores. (*Ver* ASCII)

EDP *Electronic Data Processing*, Proceso de Datos Electrónico.

Ejecutar Procesar un programa: hacer que el ordenador acepte el programa y realice las instrucciones del programa.

Ensamblador Un lenguaje simbólico de programación de bajo nivel, que utiliza la mnemotecnia en vez de las instrucciones numéricas que utiliza el código del propio ordenador.

EPROM *Erasable Programmable Read-Only Memory*. Memoria de solo lectura reprogramable. Referido también como EROM.

Ficha perforada Un medio de almacenamiento de datos en un ordenador, formado por unas fichas rectangulares de cartulina, normalmente de 80 columnas verticales cada una de ellas, donde los caracteres son codificados mediante combinaciones de orificios rectangulares perforados en ellas.

Firmware Software almacenado de una forma fija, en una ROM, PROM o EPROM. El firmware es software "afirmado" en hardware.

Floppy disk o disco blando. Diskette.

FORTRAN *FORmula TRANslation*, Traducción de Fórmulas. El primer y más ampliamente utilizado lenguaje de programación de alto nivel para ingeniería y aplicaciones científicas.

GIGO *Garbage In, Garbage Out*: Basura Entra, Basura Sale. Cuando el input es malo, el output será evidentemente malo también.

GPSS *General Problem Systems Simulation*, Simulación de Sistemas para Problemas Generales. Un lenguaje de programación de alto nivel para el desarrollo de la simulación de sistemas.

Hardcopy Visualización producida sobre papel, o sobre cualquier otro medio tangible o permanente.

Hardware Los componentes físicos y partes de un sistema informatico (los chips, CRT, cables, conexiones, teclados, paneles, luces, cintas, discos, etc.)

Hex o numeración hexadecimal. Numeración aritmética con base 16, que utiliza dígitos hexadecimales

(los diez dígitos decimales y las primeras seis letras: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F).

Hoja electrónica Programa ideado básicamente para la planificación y la predicción financiera.

IC *Integrated Circuit*, Circuito Integrado.

ID *IDentification*, Identificación.

Identificador El nombre de un elemento en un programa, como el nombre de una variable o de una subrutina.

Impresora Dispositivo periférico que ofrece el output (salida) sobre papel.

Impresora de línea Un dispositivo de salida (output) que imprime a la vez toda una línea (de 80 a 144 caracteres).

Incremento Indice por el cual se aumenta una determinada cantidad.

Indentado Arreglo de porciones de un programa o del output para mejorar su comprensión y su atractivo visual.

Inicialización El proceso de asignar valores iniciales a variables.

Input Dato entrado en el ordenador a través de un dispositivo periférico, como puede ser una lectora de fichas o un terminal.

Instrucción Una orden individualizada dentro de un programa. Puede ser una afirmación, una orden, una llamada a un subprograma, o algún otro tipo de orden que requiera que el programa realice una tarea en particular. Es la unidad fundamental de un programa de

ordenador escrito en un lenguaje de programación de alto nivel.

Instrucción de asignación Una instrucción, presente en todos los lenguajes de programación, que implica la transferencia de un valor a una variable.

Interactivo Conversacional, el caso en el que el ordenador responde cuando es activado con una pregunta adecuada.

Interface La conexión (compartida) o relación (o límite) entre dos componentes de un sistema, como por ejemplo la interface hardware-software. También: el punto donde se unen dos componentes.

Intérprete Un programa especial que traduce cada una de las instrucciones del programa del usuario (escritas en un lenguaje programador de alto nivel) a lenguaje máquina a medida que es entrado el programa, ejecutando cada instrucción inmediatamente.

I/O *Input/Output,* Entrada/Salida.

ISAM *Indexes Sequential Acces Method,* Método de Acceso Indexado Secuencial, una de las distintas formas en que los datos pueden ser organizados en archivos. (*Ver también* SAM).

Iteración Bucle, repetición.

JCL *Job Control Language,* Lenguaje de Control de Trabajo. Un lenguaje utilizado para conseguir que el ordenador realice tareas de sistema, como por ejemplo cargar un programa en la memoria principal.

Joystick Palanca de mando. Una palanca situada en o cerca del teclado de una terminal. Moviéndola en cualquiera de los cuatro sentidos se mueve un punto en la pantalla de la terminal.

K Abreviatura para $1024=2^{10}$. Así, una memoria de 16K tiene normalmente 16K bytes = 16384 bytes de memoria. Observen que, debido a que 2^{10} corresponde aproximadamente a 1000, K puede ser utilizado para indicar a grosso modo un factor de 1000.

Keyword Palabra reservada. Un nombre que aparece normalmente en un lenguaje de programación y que está expresamente reservado a algún propósito único.

Lectora de fichas Un dispositivo que entra en el ordenador la información almacenada en fichas perforadas.

LED *Light-Emitting Diode,* Diodo Emisor de Luz. Tecnología común utilizada para mostrar caracteres a través de una luz brillante (normalmente roja o verde) en una zona de lectura de las calculadoras electrónicas.

Lenguaje de alto nivel Un lenguaje utilizado para programar ordenadores que incorporan instrucciones en inglés y anotaciones matemáticas. APL, BASIC, COBOL, FORTRANS, LISP, PASCAL, PL/I, y muchos otros, son ejemplos de lenguajes de alto nivel.

Lenguaje máquina Lenguaje de programación de bajo nivel, específico para cada ordenador, y consistente en hileras de ceros y unos. Llamado también código binario.

LISP *LISt Processor,* Procesador de Listados. Un poderoso lenguaje de programación de alto nivel utilizado habitualmente en inteligencia artificial.

Lista Una estructura de datos: una secuencia de ítems de datos.

Logging on or in El acto de iniciar una sesión en la terminal de un ordenador, o de alertar al ordenador del inicio de un próximo trabajo.

LSI *Large Scale Integration*, Integración a Gran Escala. Tecnología en la cual muchos miles de componentes electrónicos se hallan integrados en un simple chip de silicio.

Mainframe Unidad Central de Proceso. *Ver* CPU.

Memoria La parte de un sistema informático cuya función es almacenar datos o instrucciones. La memoria puede ser primaria (principal o central) o secundaria (auxiliar o periférica).

Memoria Muerta *Ver* ROM.

Memoria volátil Memoria que tan sólo es conservada un breve tiempo, y que no proporciona protección contra la destrucción o pérdida de los datos almacenados en ella.

MICR *Magnetic Ink Character Recognition*, Carácter de Reconocimiento en Tinta Magnética. Una técnica asociada con la impresión y lectura de caracteres utilizando una tinta magnética especial. (Vea la parte inferior de su talonario de cheques, por ejemplo).

Microordenador Un ordenador pequeño construido en torno a un microprocesador.

Microprocesador Un procesador central normalmente equipado con un solo chip LSI IC.

Miniordenador Un ordenador pequeño, más grande que un microordenador pero más pequeño que el ordenador normal con unidad central.

MIS *Management Information Systems*, Sistemas de Información de Gestión.

Mnemónico Ayuda para la memoria. Un nombre abreviado que ayuda a recordar el nombre real, más

largo. Por ejemplo, HRDWRE puede ser un mnemónico para hardware.

Modo La forma de operación: por ejemplo, modo de ejecución.

Modem *MOdulator-DEModulator,* Modulador-Desmodulador. Un dispositivo que conecta el equipo de proceso de datos a un canal de comunicaciones.

Módulo Una porción del conjunto independiente y normalmente autónoma, como por ejemplo un segmento de programa que puede ser ejecutado aparte del conjunto del programa en sí.

Multiplex Para transmitir simultáneamente dos o más mensajes o instrucciones a través de un solo canal de comunicación.

Network *Ver* Red.

OCR *Optical Character Recognition*, Reconocimiento Optico de un Carácter. Caracteres impresos con un tipo de letra especial que puede ser leído ópticamente a la vez por las personas y los ordenadores.

Octal Anotación aritmética en base 8 que utiliza ocho dígitos (0,1,2,3,4,5,6, y 7).

Off-line Fuera de línea. Método de procesado sin hallarse en comunicación directa con o bajo el control directo de la CPU.

On-line En línea. Método de procesado hallándose en comunicación directa con o bajo el control directo de la CPU.

Operación Cualquier acción definida por una simple instrucción de ordenador efectuada en un lenguaje de programación de alto nivel. Ejemplos de operaciones

aritméticas son la suma, la resta, la multiplicación y la división.

Ordenador Un sistema que consiste en (al menos) una CPU, una memoria, unidades I/O, y una fuente de energía.

Organigrama Un diagrama o diseño esquemático de los pasos que deben ejecutarse en un programa. El organigrama utiliza una serie de símbolos estándar, tales como flechas (para indicar el sentido de operaciones de control), conectores, rectángulos, y rombos, para hacer que la finalidad y las relaciones entre los distintos pasos a efectuar resulten más comprensibles visualmente.

Output Salida. Datos transmitidos por el ordenador a un dispositivo periférico, como una impresora o una terminal.

Overflow Desbordamiento. Condición que se presenta cuando son excedidos unos límites especificados. Normalmente se aplica a la capacidad de memoria de un ordenador.

Palabra Un grupo de bits colocados juntos y tratados como una unidad en una sola localización de la memoria. Una palabra consiste normalmente en al menos 4 bytes (32 bits).

PASCAL Un lenguaje de programación de alto nivel diseñado para enseñar ciencias informáticas y buenos hábitos de programación. Es un lenguaje sencillo pero rico en facilidades de escritura y estructuración de datos.

Palabra reservada *Ver* Keyword.

Pantalla La superficie visora (CRT) donde puede ser exhibida la información.

Password Contraseña. Palabra especial utilizada para restringir el acceso, por razones de seguridad, a un sistema informático.

Perforadora Un dispositivo accionado a través de un teclado que perfora los orificios de las fichas perforadas.

PC *Printer Circuit*, Circuito impreso donde se conectan los componentes electrónicos.

Periférico Cualquier dispositivo o componente que puede ser conectado al ordenador.

PL/1 *Programming Language/One*, Lenguaje de Programación/Uno. Un lenguaje de programación de alto nivel, que combina a la vez las facilidades científicas y las comerciales.

PL/M *Programming Language for Microcomputers*, Lenguaje de Programación para Microordenadores. Un lenguaje de alto nivel derivado del PL/1, y desarrollado por INTEL para sus microordenadores.

Plotter Mesa trazadora. Un dispositivo periférico utilizado para trazar líneas, diagramas y dibujos.

POS Terminal *Point Of Sale*, Punto De Venta. Un dispositivo al estilo de una caja registradora conectado mediante una interface con un ordenador que mantiene el registro de todas las transacciones.

Proceso de datos La actividad de utilizar ordenadores y otros dispositivos para tratar la adquisición, almacenamiento y manipulación de la información.

Programa Una secuencia de instrucciones, escritas en un lenguaje de programación, y de acuerdo con los requerimientos de un sistema informático específico, que dará como resultado el realizar las tareas necesarias para resolver un problema.

Programación El proceso de escribir un programa en un lenguaje de programación en particular. También, el proceso de reducir la solución de un problema a un programa.

Query Solicitud, pregunta, indagación (de información). Los lenguajes Query han sido desarrollados para tratar las bases de datos.

Queue Cola. Una estructura de datos: una lista de ítems de datos, en cuya lista las eliminaciones son efectuadas en la cabeza (frente) y los añadidos en la cola (espalda). Una queue es una lista First-In-First-Out (FIFO) o Last-In-Last-Out (LILO).

RAM *Random Access Memory*, Memoria de Acceso Aleatorio (Memoria LSI de Acceso Directo Lectura/Escritura)

Red Un conjunto de dispositivos interconectados entre sí a través de canales de comunicación.

Registro Un conjunto o bloque de ítems de datos, estructurado en campos. Un componente de un archivo.

Robot Un dispositivo equipado con un ordenador incorporado y el equipo periférico necesario, y capaz de llevar a cabo tareas específicas y de efectuar las necesarias decisiones independientes para realizar esas tareas, pero sin intervención humana.

ROM *Read Only Memory*. Memoria Muerta, memoria permanente de un ordenador, que sólo puede ser leída, no cambiada.

RPG *Report Program Generator*, Generador de Programa de Listados. Un lenguaje especial de programación en el cual las hojas de especificaciones son llenadas para producir informes de gestión comercial. Típico de IBM

Run Procesar. Ejecutar un programa.

Salto Ramificación, desvío, salto o transferencia. Cambia el orden de ejecución de una secuencia de instrucciones, debido a una condición que requiere ese cambio. Normalmente, las instrucciones son ejecutadas en orden secuencial, una detrás de otra; pero una cierta condición puede requerir un cambio en este orden de ejecución, y el resultado será un salto.

SAM *Sequential Access Method,* Método de Acceso Secuencial. (*Ver* ISAM)

Scroll El movimiento vertical (arriba y abajo) de la información sobre una pantalla.

Secuencia comparativa El orden el que es establecida una lista de ítems, como por ejemplo en orden numérico ascendente (para números) o en orden alfabético (para nombres)

Serie Un conjunto de ítems relacionados, normalmente variables, agrupados juntos bajo un mismo nombre.

SIMULA SIMSCRIPT Lenguajes de programación de alto nivel utilizados en aplicaciones de simulación.

Sistema El conglomerado de dispositivos, gente, datos, métodos, prácticas, y todo lo demás que pueda ser necesario para alcanzar determinados objetivos.

Sistema operativo (*OS, Operating System*). Software que controla la ejecución de los programas de un ordenador. Son ejemplos: *Disk Operating Systems (DOS),* Sistema operativo orientado a disco; *Time-Sharing Operating Systems (TSOS),* Sistema operativo orientado a tiempo compartido.

Sistema gráfico Hardware y software que permiten al ordenador trabajar con gráficos y dibujos.

SNOBOL *StriNg Oriented symBOlic Language*, Lenguaje Simbólico Orientado en Cadena. Un lenguaje de programación de alto nivel utilizado en aplicaciones de procesado de cadenas, como el trabajo con textos voluminosos.

Software Los programas utilizados por un sistema informático.

Subprograma Una porción de un programa.

Subrutina Un grupo de instrucciones que pueden ser tratadas como una unidad y "llamadas" cuando son necesitadas por el programa principal.

Tabla La disposición de datos en hileras y columnas.

Tabulación El acto de disponer los datos para formar una tabla.

Teclado La parte de un terminal que contiene las teclas. Los teclados de los terminales son muy parecidos a los teclados de las máquinas de escribir eléctricas. Teclear significa entrar datos en un ordenador utilizando un teclado. La acción de apretar una sola tecla es "pulsar".

Terminal Un dispositivo periférico a través del cual una persona puede comunicarse con un ordenador, y que puede ser utilizado para enviar y recibir información.

Tiempo compartido Un método de procesado en el cual el ordenador es compartido (o, debido al tiempo muy rápido de sus respuestas, parece que es compartido) por distintos usuarios simultáneamente.

Tiempo de procesado El tiempo que necesita un programa para completar las tareas que se supone tiene que realizar.

Tiempo de respuesta El tiempo que necesita un sistema en particular para reaccionar a la señal de preparado.

Tiempo real Método de procesado en el cual el ordenador responde a, y sus respuestas afectan, la actividad que se halla bajo el control del ordenador. La guía de misiles es un ejemplo de una aplicación de computación en tiempo real.

Toro Memoria magnética a base de anillos de ferrita cableados. Cuando la corriente eléctrica pasa por los hilos, los anillos son magnetizados en una u otra dirección, y los dos estados magnéticos son representados por 0 y 1. La memoria de toros de ferrita es conocida también como memoria principal, básica o primaria (o almacén), y no es volátil. Hoy en día ha sido casi completamente desplazada por las memorias RAM, etc.

Transacción Cualquier actividad comercial, como una venta, un desembolso, una compra, un envío, una reserva, o una investigación.

Tratamiento de textos *Ver* WP.

Turnaround Time El tiempo real entre someter un trabajo (programa y datos) a un ordenador y recibir el output.

Unidad central La parte del ordenador que alberga el procesador central, o el propio procesador central como módulo independiente. *Ver* CPU.

UPC *Univeresal Product Code,* Código Universal de Producto, conocido también como Código de Barras. Compuesto por 10 dígitos, es utilizado en la mayor parte de los productos, y su objetivo es agilizar el procesado de las ventas y de las existencias de los mismos.

Utility Utilidades. Software preparado por un constructor de ordenadores para conseguir que sus productos posean una mayor eficiencia en las operaciones en tareas de rutina.

Valor Una constante, o la cantidad que ha sido almacenada en una variable.

Variable Un objeto que posee un nombre (una localización en la memoria), y que contiene o almacena un valor particular en un momento particular.

VDT *Video Display Terminal,* Terminal de Pantalla. Un dispositivo parecido a una pantalla de televisión con un teclado.

Vendedor Una compañía que vende o proporciona hardware, firmware, o software.

VLSI *Very Large Scale Integrated circuit,* circuito Integrado a Muy Gran Escala.

WP *Word Processing,* Tratamiento de Textos. El sistema informático específicamente diseñado para escribir y editar textos: formatear cartas, informes, y libros.

INDICE

EN ESTA MISMA

COLECCION

IBM PC
Steven Manus

El ordenador personal PC de IBM es uno de los microprocesadores más populares. El IBM PC es reconocido no sólo por sus aplicaciones comerciales, sino también por sus otras múltiples aplicaciones. Este libro introduce al principiante en el terreno del IBM PC y del IBM PC XT, al tiempo que informa de modo preciso acerca del funcionamiento de ambos sistemas.

- Sistema y manejo
- Teclado y pantalla
- Registro, memorización y recuperación de datos
- Aparatos periféricos y aplicaciones

COMMODORE 64
Michael Boom

El Commodore 64 pertenece a la nueva generación de ordenadores personales que ofrecen un elevado rendimiento. Ocupa una posición líder en el mercado español. El principiante podrá iniciarse en profundidad en las aplicaciones y peculiaridades del Commodore 64, familiarizándose con el funcionamiento del mismo.

- Sistema de manejo
- Teclado y pantalla
- Registro, memorización y recuperación de datos
- Aparatos periféricos y aplicaciones

ORDENADORES PERSONALES
Carlton Shrum

En este libro se exploran las partes integrantes de un ordenador personal. Se comentan las distintas posibles aplicaciones de los ordenadores personales en el mundo en que nos movemos. Facilita también la decisión que debe tomar el comprador frente a la gama de ordenadores que ofrece el mercado informático. Se trata de un libro de fácil comprensión y de gran utilidad para una buena decisión.

- Iniciación a los ordenadores personales.

BASIC

El Basic es el lenguaje de programación más importante que existe. Resulta fácil de aprender y es de aplicación universal.

El que desee utilizar un ordenador y escribir su propio programa, deberá aprender a «hablar» este lenguaje.

Mediante este libro se puede dominar el Basic a través de un método preciso, bien estructurado y de sencilla comprensión.

El principiante se convertirá paso a paso en un buen conocedor del Basic siguiendo los ejemplos de programación que aquí se ofrecen.

- Variables literales y variables numéricas
- IF... THEN... instrucción
- PRINT USING
- Bucles FOR... NEXT
- Tabla de conceptos y ejemplos prácticos

00100
9 788401 401015